

# Recovering From Errors in Conversational Dialogue Systems

B010274

MSc Speech and Language Processing

The University of Edinburgh

2011

## **Declaration**

I have read and understood The University of Edinburgh guidelines on Plagiarism and declare that this written dissertation is all my own work except where I indicate otherwise by proper use of quotes and references.

## **Abstract**

Spoken dialogue systems can encounter different types of errors, including non-understanding errors. This is where the system realises the user has spoken, but does not understand their utterance. Strategies for dealing with this kind of error have been proposed and tested in the context of slot-filling systems, for example by Dan Bohus with a system which helps reserve conference rooms [1]. However there has been little work into possible strategies for more conversational settings. This dissertation looks at how we could recover from non-understanding errors experienced by a robot tourguide, and tests the strategies in an experimental study. The main hypothesis of this study is that it is beneficial to use strategies which are designed to do something smarter than just asking the user to repeat themselves. The strategies implemented are motivated by the findings of work done on task-based dialogue systems [1, 2, 3], which suggest it is useful to move the user on through the dialogue instead of getting caught up with the non-understanding error.

## Acknowledgements

I would like to sincerely thank Colin Matheson for supervising me through this project. He has been an excellent supervisor, and I have very much enjoyed working with him. His advice has been incredibly valuable, as has the work he put in personally to make sure that the dissertation was progressing well.

I am very grateful for the expert input of Amy Isard and Jon Oberlander. Simon King also deserves special thanks for his guidance during the year.

I am grateful for all those who participated in the experimental study.

Lastly I would like to thank my family for having the patience to help me debug the system presented in this dissertation as it was being developed.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Literature Review</b>	<b>9</b>
2.1	Dialogue System Theory . . . . .	9
2.2	Detecting Errors in Dialogue Systems . . . . .	12
2.3	Sources of Non-Understanding Errors . . . . .	13
2.4	Error Handling Strategies . . . . .	13
<b>3</b>	<b>The Robot Tourguide Dialogue System</b>	<b>17</b>
3.1	Requirements . . . . .	17
3.2	Domain . . . . .	18
3.3	INDIGO . . . . .	19
3.4	TrindiKit Dialogue Management . . . . .	20
3.4.1	Speech Recognition . . . . .	22
3.4.2	Speech Synthesis . . . . .	24
3.4.3	Error Strategies . . . . .	24
<b>4</b>	<b>Experimental Study</b>	<b>29</b>
4.1	Error Strategies Implemented . . . . .	29
4.2	Experimental Hypotheses . . . . .	33
4.3	Experimental Setup . . . . .	34

4.4	Results . . . . .	37
4.4.1	Graphs of Results . . . . .	38
4.4.2	Significance Tables . . . . .	40
4.5	Conclusions . . . . .	41
4.5.1	Conclusions Drawn from Questionnaire Data . . . . .	41
4.5.2	Analyses of Errors . . . . .	42
4.5.3	More Observations . . . . .	43
<b>5</b>	<b>Conclusions</b>	<b>50</b>
5.1	Conclusions . . . . .	50
5.2	Future Work . . . . .	51
	<b>Bibliography</b>	<b>53</b>

# List of Figures

2.1	The Components of a Dialogue System . . . . .	9
2.2	Handling Misunderstandings . . . . .	13
2.3	Wizard of Oz Experiments . . . . .	14
3.1	Items in the System's domain knowledge . . . . .	19
3.2	Structure of the Tourguide Dialogue System . . . . .	21
3.3	Toy Example Grammar for Speech Recognition Hypotheses . . . . .	23
3.5	Dialogue Excerpt - A Confirmation . . . . .	26
3.4	Dialogue Excerpt- Series of Questions . . . . .	28
4.1	Dialogue Excerpt -Subsume Strategy . . . . .	30
4.2	Dialogue Excerpt -Subsume Split Strategy . . . . .	31
4.3	Dialogue Excerpt - Fake Strategy . . . . .	32
4.4	Screenshot of the Tourguide Dialogue System . . . . .	35
4.5	Breakdown of Error Types . . . . .	45
4.6	Analysis of Misunderstanding Classification . . . . .	46
4.7	Correlation of Questionnaire Questions . . . . .	46
4.8	Non-Correlation of Questionnaire Questions . . . . .	47
4.9	Total Number of Questions Across Groups . . . . .	48
4.10	Negative Correlation of Enjoyment with Error Rate . . . . .	49

# List of Tables

3.1	Update Rules in the Tourguide Dialogue System . . . . .	27
4.1	Summary of Strategies . . . . .	33
4.2	Participants Breakdown . . . . .	37
4.7	Significance of Comparisons between Groups . . . . .	40
4.8	Significance of Comparisons between MD off & on . . . . .	41

# Chapter 1

## Introduction

This dissertation is an investigation into dealing with non-understanding errors in the Spoken Dialogue System of a robot tourguide. In particular, the main aim of the dissertation is to find out whether it is worthwhile creating a library of error-handling strategies which go beyond the classic ‘Could you please repeat that?’ technique.

There are studies in the literature which look at this question within task-based or slot-filling dialogue systems (see Section 2.4). Whereas task-based Dialogue Systems focus on solving a problem through collaboration between the human user and the computer system, the robot tourguide domain is more conversational; there is no specific set of tasks to be completed other than conducting an enjoyable conversation. We will call such a dialogue system, whose only real aim is to hold an interesting conversation, a **Non-Task-Oriented Dialogue System**. As an investigation into error-recovery strategies in a non-task-oriented dialogue system, this dissertation is somewhat novel.

Natural language systems whose goal is to be interesting to the user can be very useful in real life applications. The example given in this dissertation is a dialogue system which acts as a tourguide in an exhibition. Previous work has also looked at exhibitions, focussing on the problem of tailoring texts to the interests of users in a museum, as with the ILEX system [4]. Another relevant example is MINERVA, a robot tourguide which has operated in a Smithsonian museum. In two weeks it interacted with more than 50,000 people- but its interactions were closer to small lectures to a crowd than dialogues [5]. This study presents strategies which can hopefully be used for dialogue systems whose goal is to conduct an interesting conversation with the user.

First is presented a literature review (Chapter 2), which covers the fundamentals of dialogue system theory and studies into non-understanding errors in dialogue systems. The literature on non-understanding error strategies motivates the design of several strategies in the domain of the robot tourguide. The dissertation then presents a synopsis of how the dialogue system



used for this study was created in Chapter 3. Lastly, the experimental study is described in Chapter 4, its results are given and analysed.

# Chapter 2

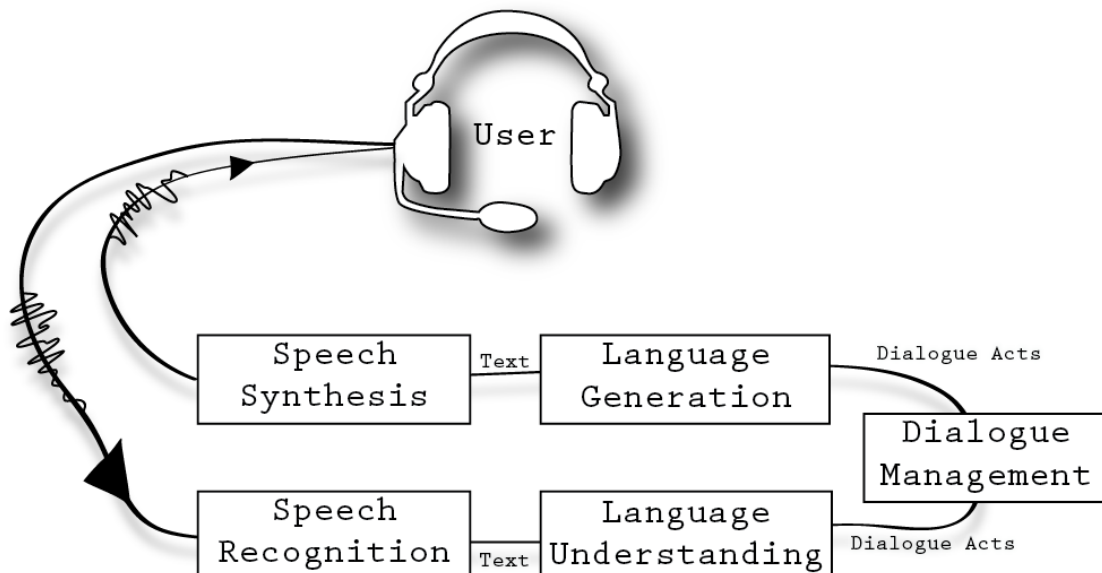
## Literature Review

This chapter contains a summary of material in the current literature relevant to this dissertation. Firstly there is an introduction to Dialogue System Theory. The Information State model is presented, which is defined formally in the paper by Larsson and Traum [6].

### 2.1 Dialogue System Theory

---

Figure 2.1: The Components of a Dialogue System



---

A Spoken Dialogue System is a computer system with which a human user can have a spoken conversation. To simplify the task of building such a system, a dialogue is assumed to be between only two agents, the user and the computer. Furthermore the dialogue is assumed to

consist of a series of **turns**. A turn consists of the user or the computer saying an utterance, then stopping.

Figure 2.1 shows the components of a Dialogue System, and what happens between the user finishing a dialogue turn and the computer starting its turn. The 5 components shown are implemented in the TRINDI Dialogue Management framework [6], in Section 3.4 it is explained how this framework is used to create a Dialogue System for use in the studies presented in this dissertation. A short review of each component in the pipeline follows:

**Speech Recognition.** The first block in the pipeline converts the waveform from the microphone into text. The state of the art is to train a Hidden Markov Model system, see Young’s review for a good overview on how this works [7]. Such a system essentially works by training up generative acoustic models of each phoneme in a variety of contexts. These models can then be used to assign a probability to a phoneme list given some acoustic data. The addition of a language model, and other tweaks such as adaptation and noise handling strategies can provide a highly effective continuous speech recognition system with a wide vocabulary.

A dialogue system can restrict the vocabulary which the Speech Recognition component expects at each turn in the dialogue. For example, the Dialogue Manager can provide a grammar which generates all possible user responses at each stage. The Speech Recognition component now has an easier task; instead of going from arbitrary speech to text, it must distinguish between just these limited possibilities. This component often also gives a confidence level with the recognised speech. It may give an  $N$ -best list with the top  $N$  hypotheses and their associated confidence levels. It can also pass on a word lattice, which condenses this information further.

**Language Understanding.** The task of this component is to convert the recognised speech to a dialogue act. A dialogue act is not the same as a full semantic interpretation, which would try to summarise all the meaning of an utterance. Rather, a dialogue act summarises the user’s intention in making the utterance. An example would be mapping “Please could we move on to the next exhibit?” to “request(move\_to\_next)”. This distinction is presented in Austin’s *How to Do Things With Words* [8], which argues that the set of sentences which state a fact/belief form only a subset of all utterances- and also important are the speech acts. The term ‘dialogue act’ comes from Searle [9], and is used to refer to the underlying action the speaker is trying to achieve with a particular utterance in the context of the dialogue.

This task can be achieved by several different methods. Broadly they can be split into hand-written and data-driven. Hand-written approaches include manually enumerating all text inputs with corresponding dialogue acts, or writing a grammar or templating system which effectively applies a series of rules to extract a dialogue act. Data-driven methods can be trained on data using Machine Learning paradigms. These include for example machine translation techniques [10].

**Dialogue Management.** This component of the pipeline accepts the user’s dialogue acts, and decides on the response of the system. Most Dialogue Management Systems employ the idea of an Information State. The Information State contains all the information the computer needs to have in its memory, which is relevant to the current stage in the dialogue. It can be thought of as the set of beliefs that the system has. The Information State is updated by the dialogue manager, and ultimately produces a dialogue act for the computer’s turn.

Again, this component can either be hand-written or data-driven. A hand-written approach essentially employs a man-made list of update rules, which interact with the Information State. A data-driven or statistical system can take into account uncertainty in the system to varying degrees.

The following definition of the Information State update procedure is essentially that of Larson and Traum [6]. In general, the Information State holds the beliefs of the system,  $b$  say- and the input from the Language Understanding component is the observation  $o$ , which contains the dialogue act given by the user with any other relevant information. Call the set of all beliefs  $\mathcal{B}$ , and the set of all observations  $\mathcal{O}$ . The update step can be seen as a mapping,  $U : \mathcal{B} \times \mathcal{O} \rightarrow \mathcal{B}$ , so the belief in the Information State is changed from  $b$  to  $U(b, o)$ . The next step is to map the new belief to a set of dialogue acts, which requires a policy mapping  $\pi : \mathcal{B} \rightarrow \mathcal{P}(\mathcal{A})$ , the powerset of all dialogue acts. Adding subscripts to index the user turns, this is summarised below:

$$\begin{aligned}
o_{t+1} &\in \mathcal{O} && \text{generated by the Language Understanding component} \\
b_t &\in \mathcal{B}; && \text{kept in memory from previous turn} \\
b_{t+1} &= U(b_t, o_t) \in \mathcal{B} \\
a_{t+1} &= \pi(b_{t+1}) \subset \mathcal{A}
\end{aligned}$$

**Language Generation.** The next step in the pipeline is to convert the system’s dialogue act(s) proposed by the Dialogue Manager into natural language. Again this can be hand-written or more automatic. An automatic approach would be to employ a Combinatory Categorical Grammar, which can start with a semantic representation or dialogue act, and generate natural language sentences. Reining in CCG Chart Realization, a paper by Michael White, presents methods to use the CCG theory in an efficient manner, fast enough for ‘the interactive needs of natural language dialogue systems’ [11]. Templates can also be a very effective (and simple) method of producing texts, providing templates of English sentences with slots where variable information can be placed. For example a template for the dialogue act  $\text{moreQ}(x)$  (where the system needs to ask the user if they would like to hear more about  $x$ ) could be “Would you like to hear a little bit more about  $x$ ?”. An even simpler method is to hand-write all the possible texts which the system could say.

**Speech Synthesis.** Most dialogue systems today would use a unit-selection type voice, which

has a large database of recorded speech. The synthesiser then solves the search task of picking optimal units in the database to minimise the perceptual effects of concatenating dissimilar units while maximising the similarity of the synthesised speech to the target [12].

**Adding more inputs.** The pipeline presented does not allow any other input or output than speech. Other inputs can be added by treating them as user moves which require another update of the Information State, just as with user moves from the speech processing components. Effectively this is increasing the dimension of elements in  $\mathcal{O}$ . Similarly, more outputs can be added by allowing the Dialogue Manager to speak to new components (adding more possibilities in  $\mathcal{A}$ ). This does not break the dialogue cycle model, where the Information State gets updated by user moves, which in turn causes outputs from the system. Other outputs that may be added include control of the movement of a robotic face, or the picture displayed on a screen interface to the user.

## 2.2 Detecting Errors in Dialogue Systems

Broadly, there are two types of errors which a Dialogue System can encounter; **non-understandings** and **misunderstandings**. A non-understanding is where the Language Understanding component fails to extract the dialogue act performed by the user. A misunderstanding is where the Language Understanding component succeeds in extracting a dialogue act, but the act extracted is incorrect.

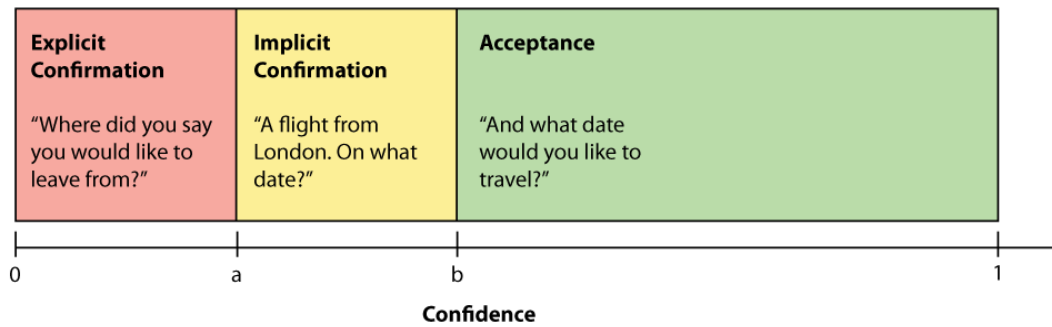
Clearly, the system will not always know if a misunderstanding has occurred, while it will always be aware of non-understanding errors having happened. Therefore detecting non-understandings is a trivial task. To detect misunderstandings, it is common to make use of confidence thresholds. Systems can use the confidence levels associated with the Speech Recognition component to prevent a misunderstanding from happening, by setting a confidence threshold below which hypotheses are rejected. This method will decrease the number of misunderstandings, but increase the number of non-understandings. The most common way of handling misunderstandings in dialogue systems is summarised in Figure 2.2. The interval  $[a, b]$  in the figure represents hypotheses whose confidence levels are too low to accept, but too high to ignore; this effectively gives a simple way for the system to detect misunderstandings. We call this the **Misunderstanding Detection interval**, and it is implemented later in this dissertation.

Note that this method of dealing with misunderstandings applies only to slot-filling or task-based dialogue systems; where there is an exchange of information between the user and the system which needs to occur. The lowest confidence threshold, between 0 and  $a$  effectively converts what was a potential misunderstanding to a definite non-understanding.

There are more complicated methods for detecting that a misunderstanding has happened at a later turn in the dialogue. For example [13] looks at detecting user turns which attempt

---

Figure 2.2: Handling Misunderstandings



---

to correct the system, utterances like "No, I said Boston- not Austin." A statistically trained decision tree using both linguistic and acoustic features is shown to be effective. (For more discussion on more advanced Misunderstanding Detection, see Section 5.2.)

## 2.3 Sources of Non-Understanding Errors

Bohus identifies the sources of non-understanding errors as falling in four categories [1]. The first is **Out-of-Application**, where the user has tried to use some functionality that the dialogue system doesn't have. Secondly is **Out-of-Grammar**, where the user has said something that the system could have dealt with if they had worded it differently, but the grammar they used was not recognised by the Natural Language Understanding component. Thirdly, there are **ASR Errors**, where the user's utterance uses grammar the system would understand and it would not be out-of-application- but the Speech Recognition component failed. Lastly there are **End-pointer Errors**, which are errors in segmenting the text into dialogue turns.

## 2.4 Error Handling Strategies

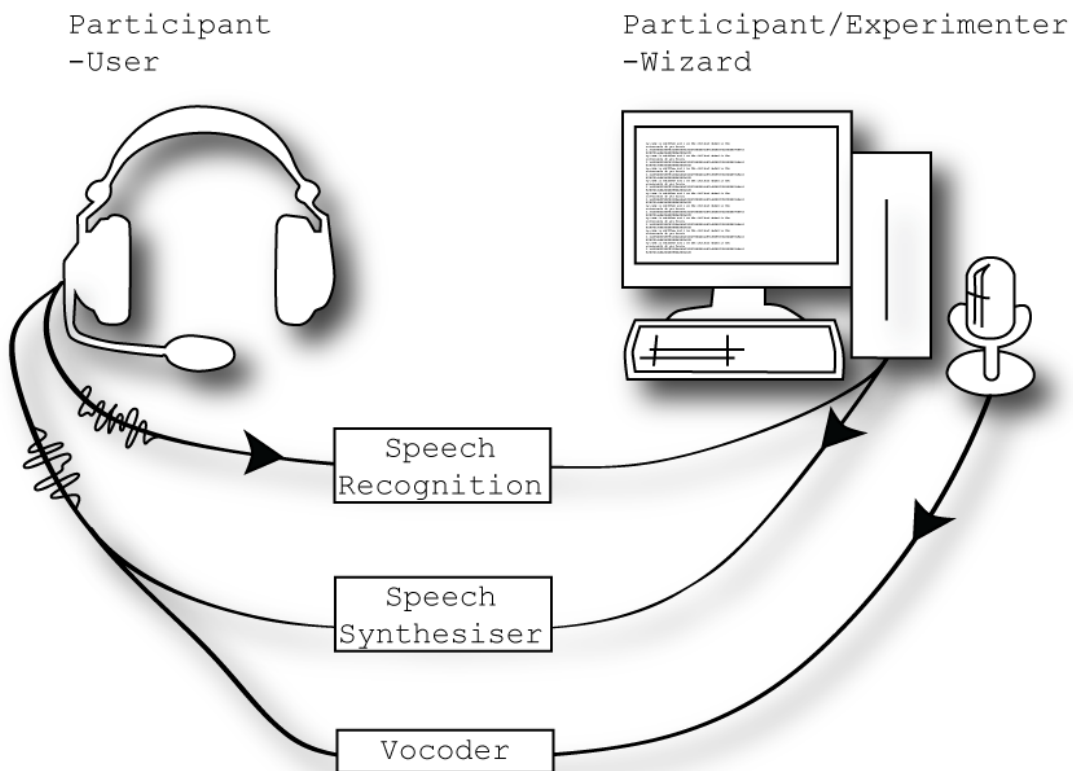
It is clearly important for the system to have a strategy for dealing with non-understandings, otherwise the Dialogue cycle will be broken, with the dialogue manager unable to update its Information State. Furthermore, non-understandings are very likely to happen in a dialogue, with current speech recognition technology. Many dialogues take part in difficult conditions, with spontaneous speech, large vocabularies, varied user populations and varying line quality [1]. Sanders et al [14] have shown that word error rates have been shown to correlate highly with user dissatisfaction in spoken dialogue systems; they also show that good error-handling strategies can help to mitigate this effect.

Once a non-understanding error has occurred in a dialogue, there is a risk that the dialogue can start on a downwards spiral. In many systems, users will typically initiate a subdialogue aimed at correcting an error whenever one appears [13]. Clearly then if another error occurs within that subdialogue, the user is starting to spiral down into a pit from where the dialogue may never be recovered. Good error-recovery strategies should therefore be found, in order to avoid this type of scenario.

There have been studies using human communication to investigate how human agents deal with non-understanding errors, with the hope this can be applied to computer dialogue systems. First was Zollo [2], who set up a Wizard of Oz experiment where one participant played the role of the dialogue system and another the role of the user. Together they had to solve problems in a virtual world, each on a different computer linked via a speech recognition layer and a speech synthesiser. The general paradigm of Wizard of Oz experiments is illustrated in Figure 2.3.

---

Figure 2.3: Wizard of Oz Experiments



---

Zollo discovered the importance of **signalling understanding and non-understanding** in human dialogues. Although only 7 individual dialogues were studied, a variety of ways of signalling understanding and non-understanding were discovered. To signal non-understandings and try to recover from them, Wizards would either ask the user to repeat themselves, or they

would ask a related question. The related question could attempt to salvage a piece of vocabulary they believed the speech recognition had got correct, or it could implicitly subsume the information the user just provided.

Asking a question which relates to what the user just said, the answer to which subsumes the answer they just gave, seems like a clever strategy. It is certainly more cunning than just asking the user to repeat themselves. An example given in Zollo's paper is:

```
user    : I am retrieve start.  (Actually: I am ready to start.)
wizard  : Are you ready to begin?
```

The answer to the Wizard's new question clearly attempts to subsume what the user just said. While asking them to repeat themselves would give away the fact that the Wizard hadn't understood, this strategy doesn't.

Zollo also found that the Wizards would deliberately signal understandings to the user, even if the speech recognition had appeared to work perfectly. This gave the users a chance to correct the Wizard in case they had actually got it wrong.

Skantze also explored how humans handle errors using a Wizard of Oz setup [3]. In this paper, 40 dialogues were analysed between a Wizard and a user. The user had to get route directions from the Wizard, to navigate a virtual world. A vocoder was used to mechanise the Wizard's voice, but the user was aware that it was a human and not a robot. Skantze emphasises in his paper the importance of asking task-related questions as a non-understanding strategy which humans use and which works well. An example of asking a task related question in the domain of this study follows below:

```
wizard  : Do you see a wooden house in front of you?
user    : Yes crossing address now.
          (Actually: I'm passing the wooden house now.)
wizard  : Can you see a restaurant sign?
```

As with Zollo's subsuming questions, this example shows the Wizard asking a question which is related, the answer to which implies the information they just missed. Skantze found that this strategy not only improved the understanding of the following utterances, but also allowed for higher user experience of task success. Clearly these are both sought after qualities of a dialogue.

Another Wizard of Oz study was conducted more recently by Koulouri and Lauria [15], in the context of direction giving dialogues. This study reveals the importance of ensuring the dialogue system has a range of rich error management resources. They also suggest that the most



successful error-recovery strategies were when the Wizard did not explicitly notify the user of an error, but instead provided relevant information and attempted to progress the dialogue.

Bohus investigated non-understanding errors and recovery strategies in a real dialogue system in his PhD thesis, *Error Awareness and Recovery in Conversational Spoken Language Interfaces* [16]. His paper with Rudnicky, [1], summarises his findings relevant to this dissertation. They used a dialogue system designed to help people book conference rooms to conduct their investigations. This is clearly a task-based domain, where there is a set of information which must be transferred from the user to the system and vice versa.

He investigated a variety of error recover strategies, of particular importance is one which he called *MoveOn*. This is where the system would continue by just asking a new question. His example is:

```
Sorry, I didn't catch that. One choice would be Wean  
Hall 7220. This room can accommodate 20 people and has  
a whiteboard and a projector. Would you like a reservation  
for this room?
```

The hope is then that the dialogue can then move on, despite the fact that the system did not understand the user's previous utterance. This strategy performed very well in relation to recovery rate, i.e. how often the following user response was correctly understood. Bohus and Rudnicky explain its success by comparing it to the other strategies, which would generally ask the user to repeat themselves, or rephrase their answer. In those cases, it is likely that the system will again be unable to understand the user's intention, as it didn't understand the user the first time. As discussed before, this process is prone to turning into a spiral of errors, where the user gets more and more frustrated. Frustration can affect the user's voice, and make the ASR less reliable. On the other hand with *MoveOn*, the system abandons the current question, and tries a new line of attack. This increases the chance of understanding the next utterance.

The *MoveOn* strategy is related to the recommendations from Zollo and Skantze, and it seems the idea of moving on, or asking a new question can be very effective. However it is not entirely clear how this strategy can be adapted to use in a non-task-oriented dialogue system. This is discussed in Section 4.1.

# Chapter 3

## The Robot Tourguide Dialogue System

In this Chapter, the Robot Tourguide Dialogue System will be described. This is the system created by the author for the purposes of investigating non-understanding error strategies. The Tourguide Dialogue System is designed to run on a robot which can act as a tourguide for visitors to the Informatics Forum, and in particular Inspace, part of the University of Edinburgh where exhibitions relating to Informatics are held.

### 3.1 Requirements

This section outlines what is required of the Robot Tourguide Dialogue System, to motivate the rest of the chapter which describes how it was created.

**Domain.** The function of the system ought to be appropriate for investigations in a more conversational context to the usual slot-filling dialogues, i.e. it should be a *non-task-oriented dialogue system*. For this reason, the task of acting as a tourguide in an exhibition was used. This allows for a very different style of conversation than with problem-solving and slot-filling dialogues (e.g. booking a flight). It must be relatively easy to write up the domain for the system, i.e. a representation of the structure of the tourguide dialogues. It should be easy to add new parts to the dialogues, including new things in the exhibition the robot can talk about. The system should be given enough information about items in the exhibit to be able to conduct a long enough dialogue with a user for the experimental study. See Section 3.2 for more information about what the system can talk about.

**Robot Ambassador.** The Informatics Forum at Edinburgh has use of an RWI B21 robot research platform, which is used as a Robot Ambassador to promote public engagement. It requires a dialogue management system so that it can interact with visitors to the Informatics Forum and the Inspace gallery using speech. This dissertation provides the robot with software which this robot can use.

**Speech.** The system should of course have Speech Recognition and Speech Synthesis components so that the user can interact with it through speech (see Figure 2.1). The synthesised voice must be at least intelligible, and the speech recognition should cope reasonably well with a wide range of voices and noise conditions.

**Interface.** There should be an interface to the system which can be displayed on a touch screen. It was decided that this should display the text which is currently being synthesised, as well as having buttons which show examples of what the user can say at each point.

**Errors.** The system must of course encounter non-understanding errors. Note that in the case of experiments, these errors do not have to be real. Rather, we can put in a new component between the speech recognition and language understanding components which sometimes throws away the hypotheses (see Figure 3.2). As well as faking non-understanding errors, the system should also be able to detect real ones which happen in the course of a dialogue. It must have access to confidence scores from the speech recognition component so it can ignore hypotheses with low scores.

During the course of the tourguide dialogues, the point where most errors are anticipated is when the system asks ‘Do you have any questions?’ The problem of dealing with the high chance that the user will ask something the system does not understand is seen as a fertile area for investigating non-understanding strategies. As well as the problem of coverage, there is a higher probability of confusion, given the ASR component will have many competing hypotheses at these points in the dialogue.

**Error Strategies.** When it encounters a non-understanding error, the system should have a selection of strategies to choose from to progress the dialogue. It should also be possible to restrict the selection of strategies available to the system, so that the use of different strategies can be compared.

**Locomotion.** The robot onto which the system is to be installed has the ability to move around the floor of the exhibition. As such, the dialogue manager should have an internal representation of where it is at every time. During the course of the experiments described in this dissertation, the system will be navigating a virtual world installed on a laptop computer- while the ultimate goal is to have a dialogue system which can communicate with a motion planner.

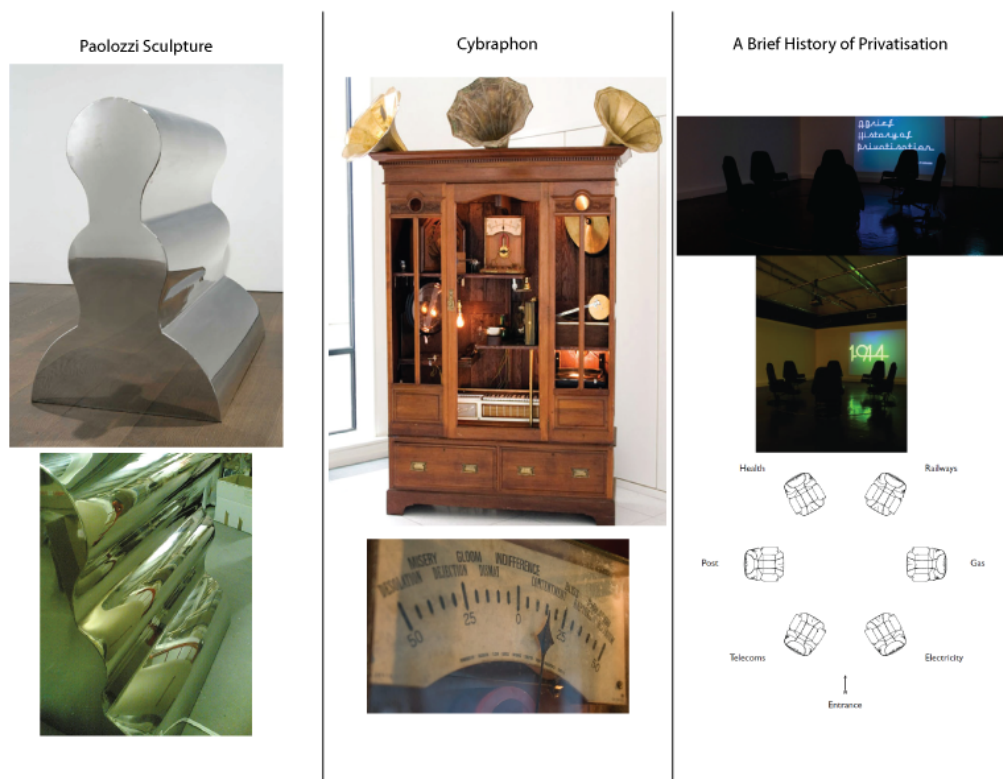
## 3.2 Domain

As discussed in the Requirements, the Tourguide Dialogue System needs to be given information about enough items in the exhibit to conduct a sufficiently long dialogue with participants in the following Experimental Study (Chapter 4). To this end, three items currently housed in Inspace and the Informatics Forum were chosen:

- A sculpture by Eduardo Paolozzi, which sits on the ground floor of the Informatics Forum.
- A robotic band-in-a-box, which plays music that is influenced by online opinion of itself. This item is called ‘Cybraphon’, and was created by an Edinburgh-based group called FOUND.
- A contemporary art exhibit called ‘A Brief History of Privatisation’ by Ellie Harrison.

These items are shown in Figure 3.1.

Figure 3.1: Items in the System’s domain knowledge



### 3.3 INDIGO

The INDIGO project (Interaction with Personality and Dialogue Enabled Robots)<sup>1</sup> was a research project aiming to advance human-robot interaction by focussing on allowing robots to perceive natural human behaviour [17]. Oberlander, J., Matheson, C. and Isard, A. of the

<sup>1</sup>See <http://www.ics.forth.gr/indigo/>

University of Edinburgh were involved in this project. This meant that the author of the dissertation had access to the system created by INDIGO, which included a dialogue manager with speech components.

The INDIGO dialogue system does however have a lot of components which are not relevant to the requirements listed in Section 3.1. As well as the dialogue system, it has a variety of other modules which all connect to an ORCA communication server, which also serves as a robot simulator. These modules include a personality model, a representation of domain knowledge using an OWL ontology, a user personalization server, and a Natural Language Generator. The robot could also interpret user gestures, and create natural gestures itself.

As a base to create the Tourguide Dialogue System, the INDIGO system was stripped down to only the speech recognition and synthesis components, the touchscreen interface, and the code which interfaces this to Prolog. The rest of this chapter describes how this skeleton was developed into the Tourguide Dialogue System.

### 3.4 TrindiKit Dialogue Management

TrindiKit is a framework written in Prolog for writing dialogue systems [6]. The framework is very much centred around the idea of a formal definition of the Information State (see Section 2.1). A developer must define what variables the Information State consists of, and then writes a system which works with these variables. The variables may be for example lists, propositions, integers etc.

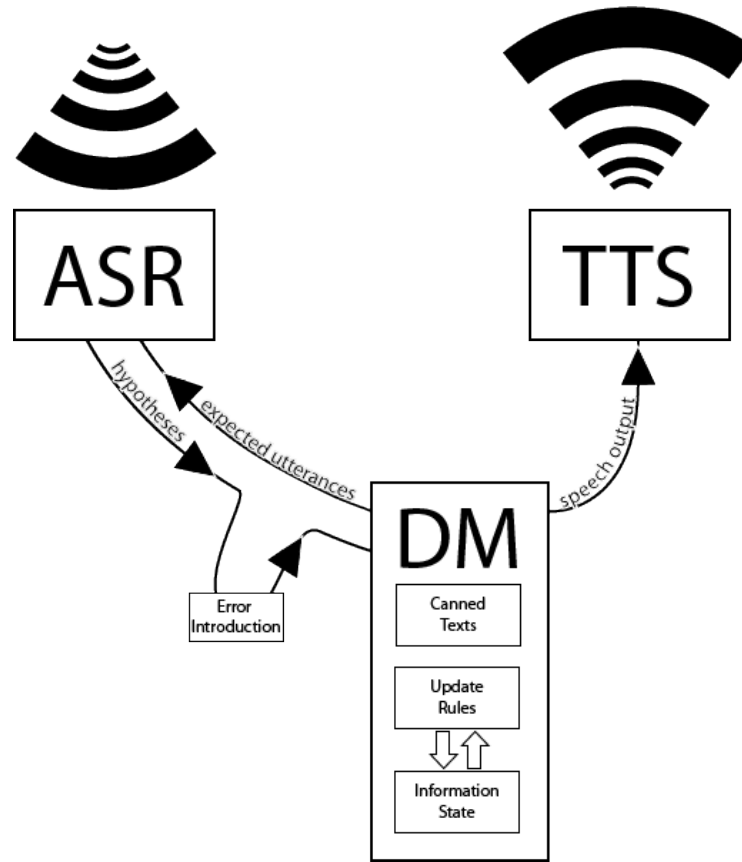
The developer writes a **set of dialogue moves** which can trigger the updates of the Information State. These will be in correspondence with natural language input from the user. They will also create a set of **update rules** that are checked for applicability in every dialogue cycle. If they are applicable, then they specify a set of consequences which act on the Information State.

At each turn in the dialogue cycle, a control algorithm is also given which specifies how to interpret input as a subset of the possible dialogue moves, which update rules to try, and how to output any system dialogue moves as e.g. synthesised speech. In general, there will be a slot in the Information State where the update rules can put a queue of dialogue moves that the system must synthesise. The speech output module can then look in this slot for things to deal with.

Figure 3.2 shows an outline of the structure of the Tourguide Dialogue System. In the Dialogue Management box is illustrated the interaction between the Update Rules and Information State, which is the core of the TrindiKit framework.

---

Figure 3.2: Structure of the Tourguide Dialogue System



---

Generic and reusable update rules were written, so that a separate domain description could be written, independent of these update rules. The domain of the dialogue, taking a user through an exhibition and allowing them to ask questions, was modelled as a graph structure. Formally the domain is defined by:

- A set of nodes,  $V$  - these represent points in the dialogue
- A set of *triggers*,  $T$  - these correspond to events which allow for the transition between one node and the next.
- A variable to hold the *current trigger*,  $t_c \in T$ , for all points in the dialogue. This is in the Information State, and holds the most recent trigger generated.
- A function,  $\phi$ , mapping each  $v \in V$  to a full description of the node, which includes everything the update rules need to know about the node. This includes e.g. what system dialogue act is associated with the node, what  $t_c$  can trigger the node, what triggers are generated by the node under different conditions, the name of the node etc. See below for a description of the node-types implemented.

Given this domain, the update rules are then written to manage the transition between nodes. There are two main types of node implemented:

**Speak and Answer** nodes. These consist of a trigger, a dialogue act which the system says when it arrives at the node, and a set of questions which the system can then answer. When the system reaches such a node, it will say the text associated with the dialogue, which at the end prompts the user if they have any questions. The system then listens out for the questions specified, and can then answer these questions (for more information see next two sections on Speech Recognition & Speech Synthesis.) When the user asserts they have no more questions, the node then generates a trigger, `leave(Nodename)` where `Nodename` is the name of the node. (Another node will then be triggered by this.)

**Split** nodes. These allow for the dialogue flowchart to split into different directions depending on user input. These must specify a dialogue act which the system uses if it enters the node, as well as a list of possible user responses with corresponding triggers. An example use of this type of node is to ask the user if they would like to hear more about an item in the exhibit. The user actions and triggers may then be: `assert(yes) → request(more, item)` and `assert(no) → request(no_more, item)`. (Then the `request(more, item)` trigger might activate a Speak and Answer node which tells the user more about the item, and gives them a chance to ask more questions.)

This structure of nodes and triggers allows the system developer to keep the set of update rules separate from the domain, which contain the details and layout of the dialogues. This also means that it is very easy to add new sections of dialogue to the system's repertoire, including new items in the exhibit it can talk about, and new behaviours. Table 3.1 lists the update rules used by the system, and thus gives a good overview of how the whole thing works.

### 3.4.1 Speech Recognition

Acapela Group<sup>2</sup> was involved in the INDIGO project, and provided the Speech Recognition and Synthesis software. Therefore there were modules already designed to be used with TrindiKit which were available to the author.

The Acapela Speech Recognition software ran on the same computer as the rest of the dialogue system, and connected to the Orca communication server. It accepted XML sent to it from the DM, to specify what it should be doing. It could either be turned off, or turned on and trying to recognise speech. When recognising speech, the XML sent to it would specify a grammar, which enumerates all possible user utterances. A more complicated system might make use of grammar rules to automatically generate speech recognition hypotheses, as the toy example in Figure 3.3 shows. Such a grammar allows for combinatorial generation of sentences, without

---

<sup>2</sup><http://www.acapela-group.com>

having to manually list all the possible options. The Figure also illustrates the problem of overgeneration, where there are utterances in the grammar which do not make sense; this could lead to another source of non-understanding errors.

In the Tourguide System, all possible user inputs were written by hand. The list specifies what natural language utterances can be expected at each node, and what dialogue act they are mapped to. Prolog allows for a reasonably compact representation of this information, by defining a predicate whose instances hold this information. Therefore for every node, we can generate a list of expected utterances, by just filtering the hand-written list.

Confidence levels are sent by the ASR component to the interpret module of the dialogue manager written in TrindiKit. As shown in Figure 2.2, if the confidence level for the top item in the  $n$ -best list is below a certain value, we call this turn a non-understanding. If the confidence level falls within a second interval, we call it a potential misunderstanding and the dialogue system will then ask for confirmation.

---

Figure 3.3: Toy Example Grammar for Speech Recognition Hypotheses

$$\begin{aligned} S &\rightarrow NP VP \\ NP &\rightarrow P \\ P &\rightarrow "I" \\ VP &\rightarrow V NP \\ V &\rightarrow "would like" \\ NP &\rightarrow DET N \\ NP &\rightarrow DET ADJ N \\ DET &\rightarrow "a" \\ DET &\rightarrow "an" \\ N &\rightarrow "icecream" \\ ADJ &\rightarrow "strawberry" \\ ADJ &\rightarrow "chocolate" \\ N &\rightarrow "lollipop" \end{aligned}$$

Generates e.g. “I would like an icecream”, “I would like a strawberry icecream”, “I would like a chocolate lollipop” etc. without having to enumerate all the possibilities.

However, it also generates “an chocolate lollipop would like a icecream”; i.e. *overgenerates*.

---

The list of hypotheses is also used to generate a list of buttons to display on the screen (see Figure 4.4). For each unique dialogue act, the first utterance which is mapped to that dialogue act is displayed as a button. This means the user can see one possible phrasing of each possible dialogue act, though there are usually many alternative phrasings written in to the system.



The ASR component is switched off while the system is speaking, so as to avoid it recognising its own speech. Furthermore, a headset with a built in microphone is used- this lowers the effect of background noise compared to using say a desktop microphone.

### **3.4.2 Speech Synthesis**

The Acapela speech synthesis component is used. The system speaks with a voice called ‘Peter’, a unit-selection proprietary voice based on the recordings of a British male voice talent. The whole pipeline of converting from text into speech is handled within this separate component, which runs alongside the other software on the same computer as the dialogue manager. It is possible to send the synthesis component data from TrindiKit.

It would be possible to do something more complicated in terms of Natural Language Generation, as alluded to in Section 2.1, but this system just uses a set of canned texts. These are pre-written English sentences, which are associated with every dialogue act which the system can perform. In fact, the use of canned texts is appropriate for this problem. It is perfectly tractable to write all the texts the system will ever say by hand, and so hardly justifiable to invest time in creating an NLG component. (If the system was required to dynamically create texts with a high degree of variation, this might be different.) Again, the texts are stored in a Prolog file using a predicate which pairs dialogue acts with the language strings.

To summarise the speech components, the system essentially has a database of canned texts for input and output- all written by hand. At each turn, the dialogue manager matches the top dialogue act in its agenda with an English string. It then sends this string to the speech synthesiser, and turns the ASR off. Once the ASR is complete, the dialogue manager looks to see which input canned texts are applicable for the current node and sends all these possible utterances to the ASR component, which starts trying to recognise them in the microphone feed. Once it has recognised a user response, the dialogue manager updates, and the cycle repeats.

### **3.4.3 Error Strategies**

The majority of a user’s time with the Tourguide Dialogue System is spent at Speak and Answer nodes- where the system has told the user about an item in the exhibit and the user asks a series of questions of the system. To illustrate this, see Figure 3.4, which shows an excerpt of a dialogue where this is happening. (This dialogue is one of dialogues collected in round 2 of the experiments, see Chapter 4.)

These main Speak and Answer nodes present the biggest challenge to the system, as the system asks a very open-ended question to the user- “Do you have any questions about this?” The

user could then ask anything about the item which has just been described. The first and most obvious way to limit errors here is to try to write a good list of questions which may be asked at these nodes.

The author created a list of around 20 questions per exhibit, with multiple phrasings per question. These were created and amended with the help of the author's friends and family, who would help generate a list of questions by either talking to the system or answering a questionnaire. The resulting lists of questions then had a rather wide coverage of what could be asked of the system.

Even with an excellent list of possible questions, it is still quite likely that the user will ask a question which is not written into the system. We might imagine a long tail of infrequent questions which the writer of the system hadn't thought of. These questions will be one source of error, the other source coming from imperfect speech recognition. In order to deal with these errors, a method for creating error strategies was written into the dialogue manager.

Firstly, on nodes other than the main Speak and Answer nodes- a default strategy of asking the user to repeat themselves at non-understandings was employed. At these nodes, the 'Please Repeat' strategy was used when the top confidence level from the ASR component fell below a certain level. Above that level, the system took the top confidence utterance as the user's input. There was no confidence interval at these nodes for potential misunderstandings, as is suggested in Figure 2.2.

At the main Speak and Answer nodes, a library of error recovery strategies was written. When the dialogue manager comes across an error at node X, it then picks randomly from the set of error recovery strategies available at the node, say it picks strategy Y, and then generates a trigger strategy (X, Y). The error recovery strategies can then be written as parts of the dialogue flowchart, using the node model described in Section 3.4, which are activated by the strategy triggers.

Also at the main Speak and Answer nodes, the Misunderstanding Detection confidence threshold is implemented. This is an interval where the system does not treat the hypothesis as totally unreliable, nor as reliable enough to proceed without confirmation (see Figure 2.2). In this case, a confirmation strategy is employed- as demonstrated in the dialogue excerpt shown in Figure 3.5.

---

Figure 3.5: Dialogue Excerpt - A Confirmation

**User:** Where was the sculptor born?

(**ASR:** Where was Paolozzi born. [confidence in Misunderstanding Detection interval])

**System:** Were you interested in finding out where Paolozzi was born?

**User:** Yes please.

[...]

---

The example in Figure 3.5 demonstrates a phrasing of a question which isn't in the system's list. However a similar question is detected by the ASR component, but at a confidence level which falls in the Misunderstanding Detection interval. The system then asks if the user had indeed asked this question.

Setting up this strategy requires writing a text for each of the questions the system can recognise. This confirmation text asks whether the user asked the particular question. Some real examples are "Were you interested in finding out where Paolozzi was born?", "Did you want to know about the music it plays?", "Do you want to know who Ellie Harrison is?".

Table 3.1: Update Rules in the Tourguide Dialogue System

Rule Name	Function
getLatestMove	Serves as grounding, gets latest move from the interpret module and copies into the Information State.
incrementCounter	Increments a counter every time a question is asked. This allows the Error Introduction function to know when to simulate a non-understanding error.
failedMove	Recognises when a non-understanding error has occurred.
introduceError	Introduces an error to the system if conditions are met.
setExpectations	Fills in the expectedXML slot in the Information State with the XML for the ASR component, which contains the utterances the system the user may say next, as well as the button mappings for the touchscreen interface.
outputStored	Sends the XML stored in expectedXML to the ASR component when required.
noMoreQuestions	Detects when the user has no more questions, and creates a <code>leave_node()</code> trigger.
splitBranching	Manages the linking of user input to triggers, if the system is currently at a Split node.
goTo	Handles movement from one area in the exhibit to another.
errorRecovery	In case of a non-understanding error, this picks an error strategy to use.
clarification	Deals with asking for clarification if a potential misunderstanding has occurred.
link	Allows for rules in the domain which convert one trigger into another, given a set of conditions. Useful for example in changing behaviour of the system depending on where in the exhibition the system has been.
speakAndAnswer	Allows the system to enter a Speak and Answer node.
answerQuestion	Answers a user question if the system is at a Speak and Answer node, and they have asked a question in the node's list.
split	Allows the system to enter a Split node.
errorRecoveryDefault	Defines the default error recovery strategy for every node, which is just to ask the user to repeat themselves.
usedStrategies	Keeps a record of which error strategies have been used for each item in exhibit.
visitedLocations	Keeps a record of which items in the exhibit the user has been to with the system.

---

Figure 3.4: Dialogue Excerpt- Series of Questions

**System:** The display case playing music is called Cybraphon. It's an interactive band in a box, inspired by the nickelodeons of the nineteenth century. Just like a real band, it is influenced by community opinion. It is always searching for reviews and comments about itself online, and that affects the emotion in its performance. Do you have any questions about this?

**User:** What is a nickelodeon?

(**ASR:** what is a nickelodeon)

**System:** A nickelodeon was an automated instrument. You would put in a coin, like a nickel, and it would play a song. They were superseded by the juke box. Do you have any more questions about this?

**User:** How does it work?

(**ASR:** how does it work)

**System:** It is controlled by a hidden computer, which is running custom software. It is always checking the internet, and updating Cybraphon's emotions according to how its popularity is changing. Do you have any more questions?

**User:** What kind of music does it play?

(**ASR:** what type of music does it play)

[...]

---

# Chapter 4

## Experimental Study

Now that the Tourguide Dialogue System has been described, this Chapter presents the experimental study which was carried out to investigate non-understanding errors. There were two rounds of experiments, run with slightly different versions of the Tourguide Dialogue System.

### 4.1 Error Strategies Implemented

Recall from Section 2.4 that a promising type of non-understanding error strategy in task-based domains is to use task-based questions to move the dialogue on. In a non-task-oriented dialogue system, there is not necessarily a piece of information that the robot needs to get from the user, so it doesn't always make sense to ask a related question. In fact in the Tourguide Dialogue System, it is not the robot which is asking systems of the user but in fact the other way around.

The problem is to think of a subsuming question to progress the dialogue when the user has just asked a question about an item in the exhibit, which the system hasn't been able to understand. The first strategy which attempts to do this is called **Subsume**. Figure 4.1 shows an example of the strategy being used in one of the dialogues from the experimental study:

---

Figure 4.1: Dialogue Excerpt -Subsume Strategy

**User:** When was this sculpture made?

(**ASR:** when was the sculpture made)

ERROR INTRODUCED

**System:** Well, are you interested in finding out a little more about Paolozzi and his sculpture work?

**User:** Eh, yes.

(**ASR:** yes)

**System:** Well Paolozzi was a son of Italian immigrants, born in North Edinburgh. He spent a lot of time in Scotland, and also lived in Germany, France and the USA. Sadly he died in 2005, but his sculpture work is to be seen all over the UK. The Informatics Forum acquired a collection of his work, including 4 sculptures like this one. Do you have any more questions about this?

[...]

---

The system first asks if the user is interested in finding out a bit more about the item. It waits for a response, any response, and then proceeds to give a short text about the item. The text is designed to incorporate answers to a lot of the possible questions which the user may have asked. It is hoped that the user will find this information interesting. The system tries to broaden the user's goal from obtaining a specific piece of information, to just hearing some general interesting information about the piece. The system essentially manipulates the user into hearing the subsuming information, in place of the specific answer. The strategy is of course designed to hide the fact that the system didn't understand.

The second strategy is **Subsume Split**. This strategy is very similar to Subsume, but it gives the user a choice as to what subsuming information they prefer. The questions for every item in an exhibit should broadly be able to be split into two categories. For a sculpture, this could be a) questions about the sculptor as an individual and b) questions about this sculpture and the sculptor's work. An example of this strategy in use is shown in Figure 4.2.

---

Figure 4.2: Dialogue Excerpt -Subsume Split Strategy

**User:** What does it represent?

(**ASR:** what does it represent)

ERROR INTRODUCED

**System:** Well, are you more interested in Paolozzi himself, or his sculptures?

**User:** Eh, his sculptures.

(**ASR:** his sculptures)

**System:** Paolozzi's sculpture work was inspired by some of the great minds in the history of science. He invented a method of casting bronze sculptures, analogous to the process of creating collages. His early collage work in Paris pre-empted Pop Art. There are 4 of his sculptures in the Forum, and his work can be seen throughout the cities of Britain. Do you have any more questions?

[...]

---

This is part of the same dialogue as the previous excerpt. The thinking behind this strategy is much the same as Subsume. However, by giving the user a binary choice, it is hoped the information presented will be of more interest to the user and be more closely related to their original question. One potential drawback compared to Subsume, is that Subsume Split requires understanding the next response of the user. Whereas Subsume will take any user input and proceed, Subsume Split needs to differentiate between the two options. In reality, the ASR success rate is extremely high for such binary choices. It would be wise though to choose a question where the two natural answers are phonetically contrasting- e.g. 'Paolozzi himself' and 'His sculptures.'

The last strategy implemented is called **Fake**. This is actually a strategy which was investigated in a non-published study of the INDIGO project. The idea is for the system to pretend it has 'forgotten' to include a piece of information. The excerpt in Figure 4.3 shows this strategy being used, at a different time in the same dialogue as the previous two dialogues.



---

Figure 4.3: Dialogue Excerpt - Fake Strategy

**User:** Why?!

(**ASR:** why)

ERROR INTRODUCED

**System:** I really should have mentioned. Last year, Ellie Harrison's work 'Trajectories' was here in Inspace. That was an interactive project where you could compare your life's timeline with others.

**System:** Did you have a question about this installation, or Ellie?  
[...]

---

Although it is clearly unrealistic for a robot to forget something, it is hoped that the piece of information which the system provides is sufficiently interesting to justify saying it instead of answering the user's question.

Lastly, there is the **Please Repeat** strategy, which simply asks the user to repeat their question. This is used as a baseline strategy, to which the previous three are compared in the study. Table 4.1 summarises the strategies implemented in this study.

Table 4.1: Summary of Strategies

Name	Description
Subsume	Ask if the user would like to hear more information about the item
	E.g.: “Would you like to hear a little more about Paolozzi and his sculptures?”
Subsume Split	Ask if the user is more interested in hearing about aspect A or aspect B of the item
	E.g.: “Well, are you more interested in finding out more about Paolozzi himself, or his sculptures?”
Fake	Fake having forgotten to say something interesting about the item.
	E.g.: “I meant to add; one of Paolozzi’s most famous works can be found here in Edinburgh. At the top of Leith Walk, there are sculptures of human body parts, including a giant foot, by Paolozzi.”
Please Repeat	Just ask the user to repeat their question.
	E.g.: “Please could you repeat that? Just say ‘no’ if you have no more questions.”

It is hoped that the motivation behind each of the strategies is sufficiently clear that they might be implemented in other domains relatively easily.

## 4.2 Experimental Hypotheses

As discussed, the purpose of this dissertation is to investigate non-understanding errors in a non-task-oriented dialogue system. So what will the investigation consist of? This section describes the hypotheses behind the experiments conducted, in order to motivate their design.

**‘Users prefer the Strategies to Please Repeat’.** The first and main hypothesis is that using the Subsume, Subsume Split and Fake strategies is preferable to a user than only using Please Repeat. The literature seems to suggest it is beneficial to have a library of non-understanding errors available to the system (see Section 2.4) for task-based dialogue systems. It is hypothesised that this is also the case in non-task-oriented systems. ‘Preferable’ and ‘beneficial’ need to be re-expressed into measurable quantities. In a task-oriented system, this might mean looking at task-success. However in the Tourguide Dialogue System, there is no intrinsic measure of task success. For this reason, Mean Opinion Scoring was used, with users answering questions which assessed how they feel their interaction with the system went. This is discussed in more detail in Section 4.3.

**‘Misunderstanding detection improves the dialogue from the user’s perspective’.** A secondary hypothesis is that we can improve the user’s ratings of the dialogue by including misunderstanding detection. Recall this is when we set a confidence interval in which the system is neither sure that it failed to understand the user, nor that it understood the user correctly. In this case the system will say something like “Were you interested in finding out where Paolozzi lived?” This is very much a secondary hypothesis, the study is mainly interested in the users’ preference for the new strategies. However an insight into Misunderstanding Detection is also given in this dissertation.

As well as testing these two hypotheses, the experimental setup described in Section 4.3 will permit other analyses, such as an analysis of the different error types, how various measurements of the interaction are correlated, and how well misunderstandings are classified.

## 4.3 Experimental Setup

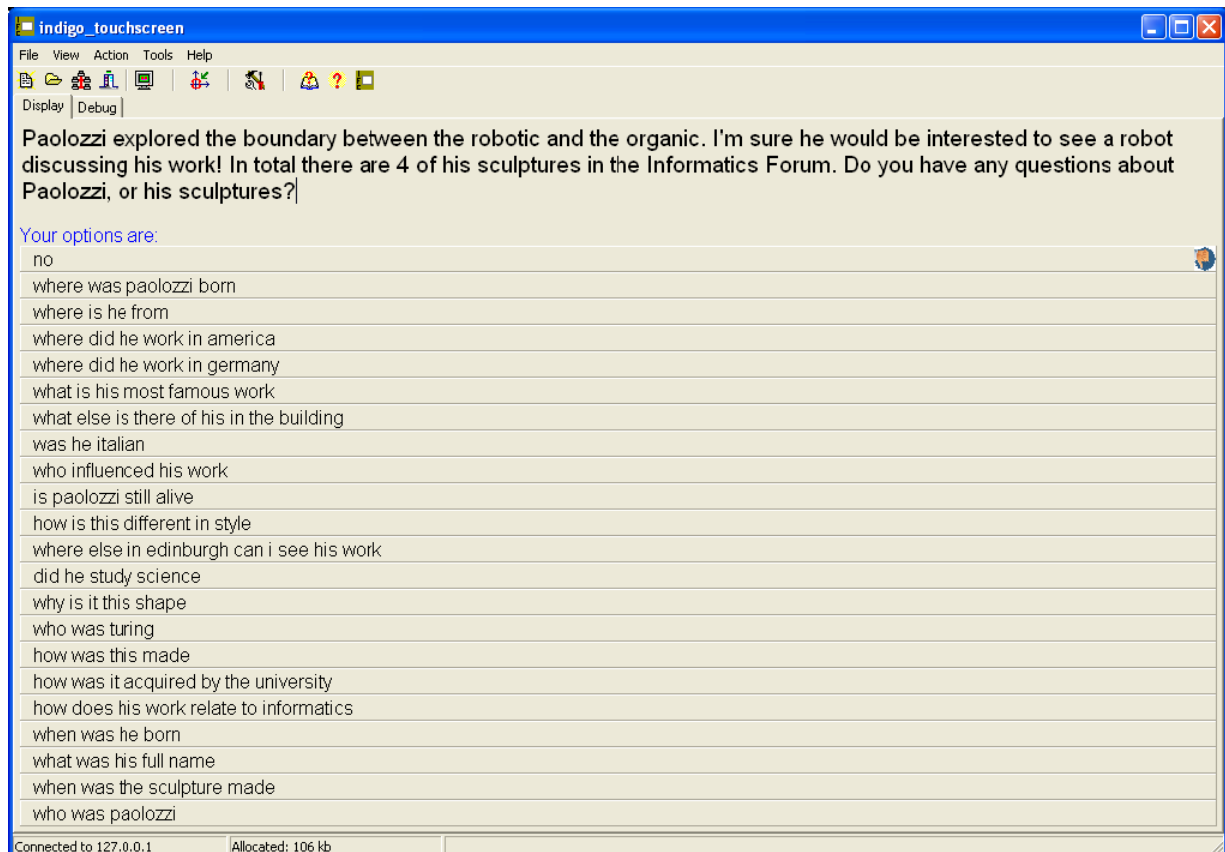
The experiment essentially consists of having participants split into two groups, and individually conducting an interaction with a version of the Tourguide Dialogue System which depends on their group. Members of Group A interact with a version which only has at its disposal the Please Repeat strategy when confronted with non-understanding errors, while members of Group B interact with a version which uses the Subsume, Subsume Split and Fake strategies. (See Table 4.1 for a description of each of these strategies.)

Participants are sat in front of a laptop running the Tourguide Dialogue System, and asked to conduct a 10 to 15 minute long dialogue. They are not given any particular tasks to achieve, so as to reflect a natural interaction with a non-task-oriented system- other than the length stipulation. To aid them in understanding what the exhibits look like, they are given a printed copy of Figure 3.1. They are told to imagine they are at these items in the exhibition with the robot, and to ask their own questions, or use the on-screen examples as inspiration.

Recall that the Dialogue System uses a touchscreen interface to display buttons for possible inputs at all times. These buttons enumerate one phrasing per question at each Speak and

Answer node, and thus aid the user in thinking of questions. Figure 4.4 shows what the screen looks like during an interaction, at one of the main Speak and Answer nodes in the dialogue. The participants are not told that the buttons enumerate all possible dialogue acts (but not all possible phrasings), but rather that they are a list of examples. This encourages them to try different questions, necessarily inducing real non-understandings.

Figure 4.4: Screenshot of the Tourguide Dialogue System



The system is set to simulate a non-understanding error at every third question asked. If a real non-understanding occurs, then the counter is reset, so it only introduces errors if they are needed.

As mentioned, participants are split into two groups; members of Group A interacting with a system which only has Please Repeat at its disposal, and members of Group B interacting with a system which can use the other strategies; Subsume, Subsume Split and Fake. The policy for choosing which strategy to use is just to pick an unused strategy at the current node at random. If the Group B system runs out of strategies at a node, it will start using the Please Repeat strategy.

At the end of their interaction, participants are asked to fill in a questionnaire. The questionnaire includes a series of statements with which the participant must specify their level of

agreement on a scale of 1 to 5. The statements are listed below, with a keyword for each one in square brackets. The keyword provides a way to quickly refer to the sentences individually in later Sections.

- The system understood what I said. [**understood**]
- My conversation with the system flowed smoothly. [**flow**]
- The system is intelligent. [**intelligent**]
- The system was helpful. [**helpful**]
- It was clear what was happening when the system did not understand me. [**clear**]
- My conversation with the system was interesting. [**interesting**]
- I enjoyed talking to the system. [**enjoyed**]
- I felt confused when talking to the system. [**confused**]
- I felt frustrated when talking to the system. [**frustrated**]

These are designed to measure the user's satisfaction with the interaction along multiple dimensions. As there is no intrinsic measure of 'task success', these questions serve to quantify how well the interaction went from the user's perspective. In designing a dialogue system, we would like to maximise the level of agreement with the first seven questions, and minimise that of the last two. (However, we may prioritise each dimension differently- for example we might want the conversation to flow smoothly, at the expense of the user's opinion of how intelligent the system is.)

So far, the experiment is adequately designed to test the first hypothesis. By comparing the questionnaires from Group A and Group B, we can test how user opinion differs between the Please Repeat strategy, and the other strategies. In order to address the second hypothesis, that misunderstanding detection is useful, the same experiment was run in two rounds. In the first round there was no misunderstanding detection, and in the second round there was misunderstanding detection.

This means we effectively have 4 groups of participants, identified first by whether they had misunderstanding detection switched on (yes or no), and secondly by their group (A or B). Comparisons should only be made between the collected results of individual groups; it would be questionable to e.g. pool the results of all Group A participants and compare them to all Group B (independent of misunderstanding detection).

As well as having questionnaire responses to analyse, the system was set up to save a transcript of its state throughout each interaction. Also, four of the dialogues in the run where Misunderstanding Detection was switched on were recorded and transcribed by hand. This meant scripts could be generated which showed what the user actually said, as well as what the system did. These scripts have been used to illustrate the previous sections, for example see Figure 3.4.

## 4.4 Results

Altogether, 30 people participated in the study. This number is broken down in Table 4.2.

Table 4.2: Participants Breakdown

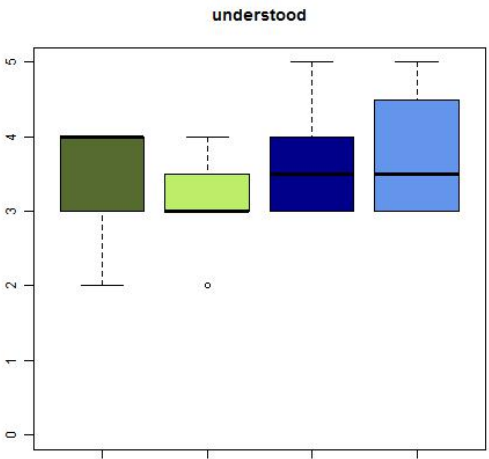
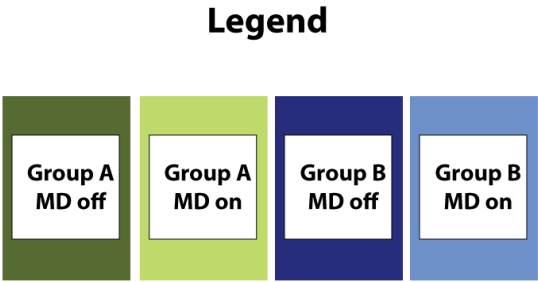
	<b>Group A</b> (Please Repeat only)	<b>Group B</b> (Subsume, Subsume Split and Fake)
<b>Misunderstanding Detection Off</b>	7	7
<b>Misunderstanding Detection On</b>	8	8

Each interaction with the system was no longer than 15 minutes, and no shorter than 10. The average interaction was 13 minutes long. The next subsection (4.4.1) shows graphs of the results for the nine questions in the questionnaire. From left to right, the box plots show Group A when Misunderstanding Detection (MD) was off, Group A with MD on, Group B with MD off and lastly Group B with MD on.

The rectangle of a boxplot shows the interquartile range, from the first quartile to the third. Values more than  $\frac{3}{2}$  times the interquartile range higher than the first quartile are treated as outliers, as are values that much lower than the third quartile. Outliers are plotted separately with an open disk. The whiskers go from the minimum to the maximum value of non-outliers. The median value is shown as a thick line.

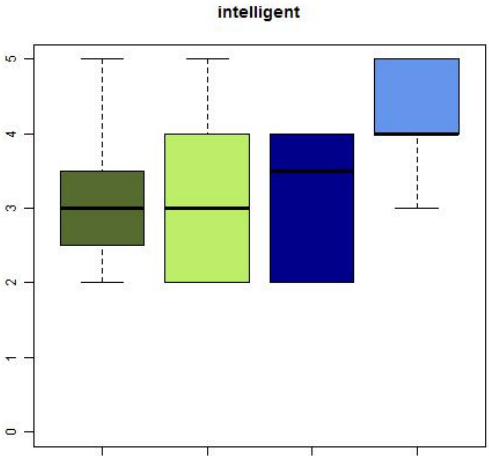
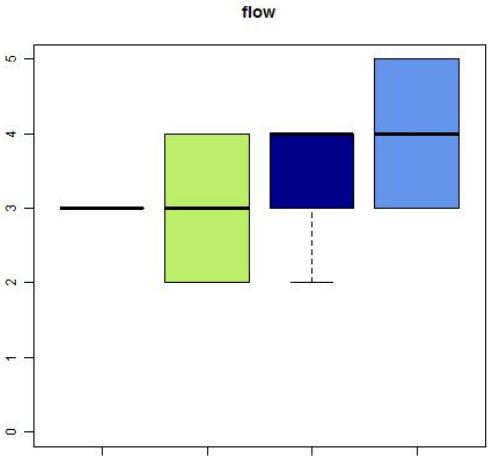
The following subsection (4.4.2) gives two tables, which summarise statistical significance tests on the results. Here we test the hypothesis that the answers to the questionnaire are actually predictable by either the Group the participant was in, or whether MD was on for them. An exact Wilcoxon rank sum test is run on R in each case to test the null hypothesis that the two sets of data come from the same distribution. In each test we are comparing 7 or 8 numbers with 7 or 8 other numbers, so it is quite hard to achieve statistical significance. Nevertheless, comparisons with  $p < 0.1 = 10\%$  are printed in bold.

4.4.1 Graphs of Results



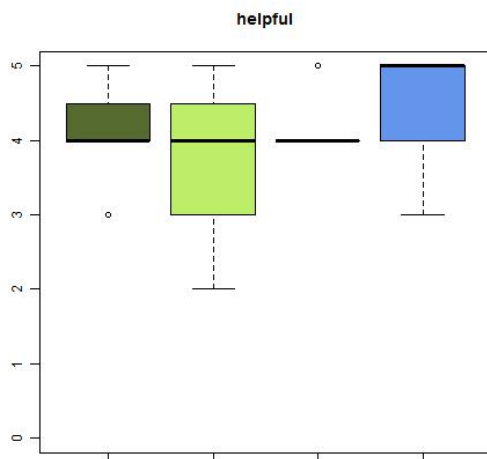
(MD = Misunderstanding Detection)

The system understood what I said

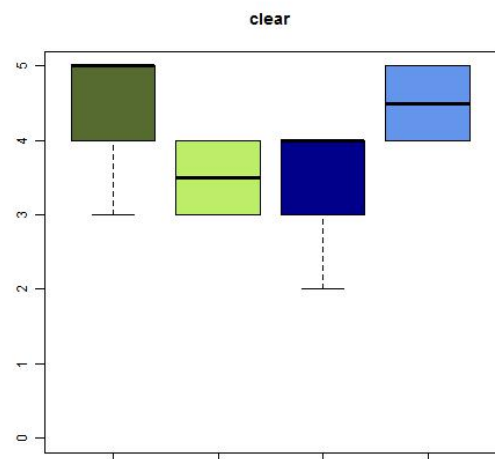


My conversation with the system flowed smoothly

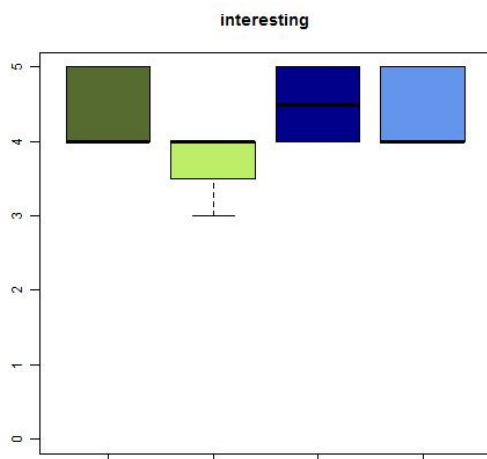
The system is intelligent



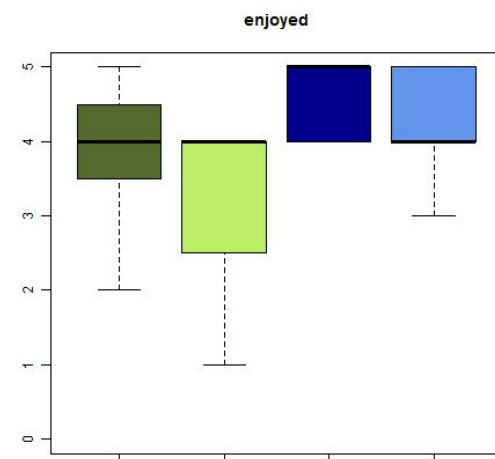
The system was helpful



It was clear what was happening when the system did not understand me

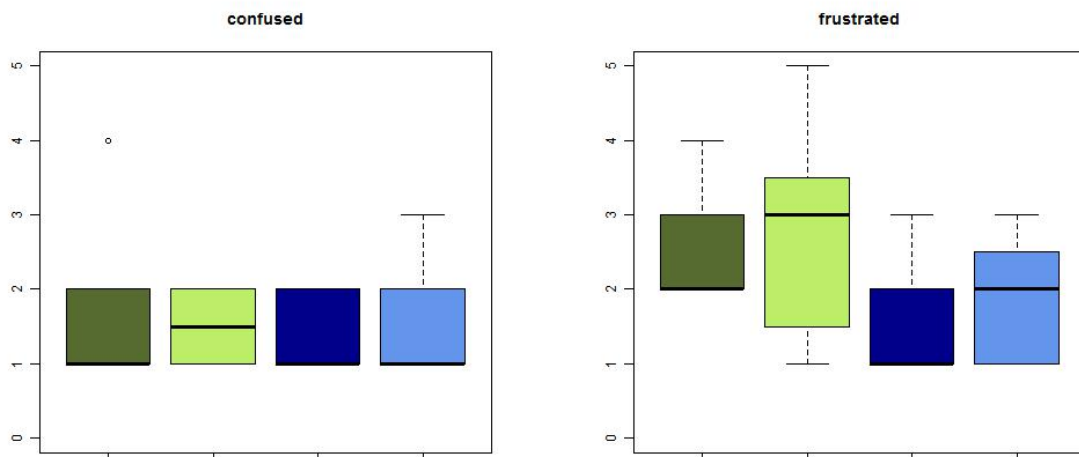


My conversation with the system was interesting



I enjoyed talking to the system





I felt confused when talking to the system    I felt frustrated when talking to the system

#### 4.4.2 Significance Tables

Table 4.7: Significance of Comparisons between Groups

	MD On					MD Off				
	Group A		Group B			Group A		Group B		
	(Please Repeat)		(Strategies)	p-value		(Please Repeat)		(Strategies)	p-value	
understood	3.13	<	3.75	22.5%		3.43	<	3.67	75.2%	
flow	3.00	<	4.00	8.8%		3.00	<	3.50	5.4%	
intelligent	3.13	<	4.25	7.6%		3.14	<	3.17	90.0%	
helpful	4.75	>	4.50	17.1%		4.14	<	4.17	100.0%	
clear	3.50	<	4.50	1.1%		4.43	>	3.50	4.5%	
interesting	3.75	<	4.38	7.2%		4.43	<	4.50	100.0%	
enjoyed	3.25	<	4.25	5.9%		3.86	<	4.67	0.2%	
confused	1.50	=	1.50	100.0%		1.20	<	1.33	100.0%	
frustrated	2.75	<	2.88	24.5%		2.57	>	1.50	3.8%	

Table 4.8: Significance of Comparisons between MD off & on

	Group A					Group B			
	MD On		MD Off	p-value		MD On		MD Off	p-value
understood	3.13	<	3.43	38.0%		3.75	>	3.67	100.0%
flow	3.00	<	3.00	100.0%		4.00	>	3.50	41.1%
intelligent	3.13	<	3.14	100.0%		<b>4.25</b>	>	<b>3.17</b>	<b>4.2%</b>
helpful	3.75	<	4.14	56.9%		4.50	>	4.17	24.2%
clear	<b>3.50</b>	<	<b>4.42</b>	<b>4.6%</b>		<b>4.50</b>	>	<b>3.50</b>	<b>3.3%</b>
interesting	<b>3.75</b>	<	<b>4.43</b>	<b>3.3%</b>		4.38	<	4.50	100.0%
enjoyed	3.25	<	3.86	35.6%		4.25	<	4.67	41.7%
confused	1.40	<	1.50	89.0%		1.50	>	1.33	83.2%
frustrated	2.75	>	2.47	82.0%		1.88	>	1.50	46.6%

(Again, MD is Misunderstanding Detection)

## 4.5 Conclusions

### 4.5.1 Conclusions Drawn from Questionnaire Data

Firstly are presented a list of conclusions which can be drawn from the raw results of the previous Section. The first set of conclusions draws from the data summarised in Table 4.7.

- Use of the new strategies (Subsume, Subsume Split and Fake) increase user **enjoyment** of the dialogues. Participants said they enjoyed the interactions roughly one point more with the strategies than without (independent of Misunderstanding Detection), and this had high statistical significance.
- The system which used the new strategies with MD on was rated as significantly more **interesting**. Similarly it was rated as less **frustrating**, but this was only significant when MD was off.
- Use of the new strategies significantly increased the user's perception of how well the **dialogue flowed**.
- The participants' opinion of how well the system **understood** them was largely independent of using the strategies. On average, they rated the system which used the strategies as understanding them better- but this difference was not significant.
- Also statistically insignificant with respect to Group was how **helpful** the system was perceived to be, and how **confused** the participant felt.

- Strangely, participants agreed with the **clear** sentence less when it used strategies and MD was off, but more when it used strategies and MD was on. (The **clear** sentence was ‘It was clear what was happening when the system did not understand me.’) Quite high significance levels were reached in these tests ( $p = 0.011$  and  $p = 0.045$ ). This suggests that there is some interaction between clarity of the system, using the strategies, and the Misunderstanding Detection. It is entirely possible that this is an anomaly in the results (we are doing quite a few comparisons, it is actually rather likely that something like this happens.)

The next set of conclusions are justified by Table 4.8, which looks at the effect of Misunderstanding Detection.

- In general, there seems to be very little effect of introducing the Misunderstanding Detection; very few comparisons prove to be statistically significant. This is most probably because switching it on and off is not a very large change in the behaviour of the system. Misunderstandings are in fact relatively rare (see Subsection 4.5.2 for an analysis of the MD component). Furthermore, the analysis suggests that the MD component’s threshold system is inadequate for classifying misunderstandings well.
- That said, when the system is using the new strategies (Subsume, Subsume Split & Fake), the system seems significantly more intelligent when it has MD switched on.

### 4.5.2 Analyses of Errors

Figure 4.5 shows a break down of what types of errors the system was aware of during the study. Note that when Misunderstanding Detection is switched on, the number of genuine misunderstandings is dramatically decreased. This is because MD often converts what would previously have been classed as a non-understanding as a misunderstanding. The average number of errors per person increases when Misunderstanding is switched on, which is because what previously would have passed as an understanding is now caught as a potential error.

Recall the threshold for MD was tuned by hand, so how well did the thresholds work? Let’s treat the MD component as a classifier, which must classify misunderstandings, distinguishing them from ‘understandings’ - i.e. times when the top item from the ASR component is indeed what the user said. We can then estimate how many false positives, true positives, false negatives and true negatives the component classified.

**False positives** are understandings which were classified as misunderstandings. The number of these can be estimated by counting how many times the user answered ‘yes’ to the misunderstanding clarification question (e.g. “Were you interested in finding out when Paolozzi

was born?”). In reality this is likely to be an overestimate, as users may decide they are in fact interested in finding out about this new information, even though they asked something else.

**True positives**, misunderstandings correctly identified, can be estimated by how many times users said ‘no’ to misunderstanding clarification questions.

**False negatives** are misunderstandings which were incorrectly classified as understandings. These can only be counted if a transcription of what the user actually said exists. As stated in Section 4.3, 4 of the 16 dialogues with MD switched on were recorded and transcribed by hand. In these 4 transcripts, the author counted occasions when the system fails to ask a misunderstanding confirmation when the system’s top hypothesis was incorrect. Multiplying this number by 4 gives an estimate for the number of false negatives in all 16 dialogues.

**True negatives**, i.e. understandings which were correctly identified, are similarly estimated as occasions in the 4 recorded dialogues where the system was correct in its first ASR hypothesis, and didn’t ask for confirmation. Again this is multiplied by 4 to get an estimated for all 16 dialogues.

Figure 4.6 shows the values of each of these numbers. We see that there is quite a lot of error in the classification. Understandings are usually quite well classified, especially remembering that the 30 figure is likely to be an overestimate (see above). However quite a lot of misunderstandings are missed by the system. The figure also suggests that a hard-coded confidence interval isn’t likely to work particularly well. Moving the threshold down would miss more misunderstandings, and moving it up would cause more understandings to be incorrectly classified as misunderstandings.

A more clever method of Misunderstanding Detection might give better results. For example Bohus and Rudnicky present data driven methods to better deal with confidence scores[18]. The general idea is to try to learn a model for each individual user instead of having a fixed hand-tweaked interval built into the system.

### 4.5.3 More Observations

In this section, some more observations on the results of the study are presented.

Firstly, it is interesting to note how answers to particular pairs of questions in the questionnaire correlate. Figure 4.7 shows how the perceived intelligence of the system and the flow of the conversation are correlated with how much the participant enjoyed the interaction. These correlations are significant according to a Pearson’s correlation test. (The size of the disk at a particular point is proportional to the number of data-points at that point, and data is taken over all 30 participants). Figure 4.8 shows that how user enjoyment is not correlated with how clear it is what the system is doing when it doesn’t understand the user. (The null hypothesis

cannot be rejected for these variables, with  $p = 0.65$ .) This implies that it is not necessarily important for the user to understand what the system is ‘thinking’ at each stage, for them to enjoy the dialogue.

Other studies do show that a good understanding of the system’s internal state can help reduce error rates, see e.g. [19, 1, 20]. Karsenty analyses the effect of using ‘transparency strategies’, which make the system ‘visible’ to the user, effectively increasing the user’s expertise [20]. These significantly decrease the rate of errors. However the importance of system transparency is likely to be dependent on the domain of the system; the cited references are all for heavily task-based systems. Perhaps it is less important for the user to understand the precise inner workings of a non-task-oriented system, as the user doesn’t need to learn how to manipulate the system to do what he or she wants.

One important difference between the Groups is the total number of questions differs quite dramatically. See Figure 4.9, recalling Group A had a system which only used Please Repeat, and Group B had a system which used Subsume, Subsume Split and Fake. Note that the number of questions asked by members of Group A is clearly higher than that of Group B. The reason for this is probably that the new strategies introduce a lot of information to the user, which they wouldn’t be exposed to with only the Please Repeat strategy. The new subsuming information answers a lot of the user’s potential questions in advance. The new strategies allow for the robot to do more of the speaking, which in turn lowers the error rate.

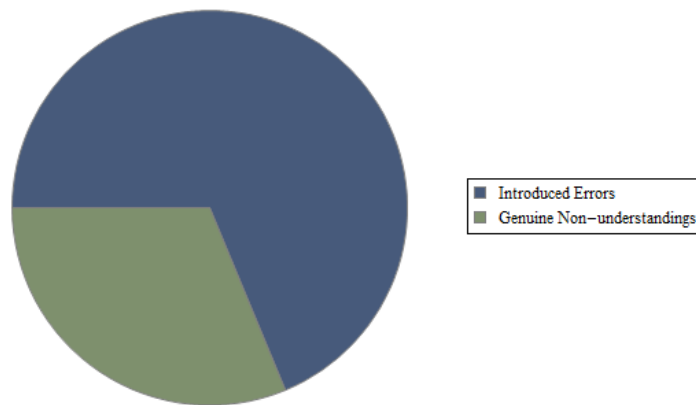
So one useful side-effect of the new strategies is that they give the system an excuse to do more talking, which is beneficial as so long as the system is talking it doesn’t need to rely on its speech recognition capability. Also, the strategies allow the system to tell the user what it is determined to be the most interesting information for the user. These interesting pieces of information potentially make the conversations more enjoyable and interesting than a conversation where the system only gives information which directly answers the user’s questions (as with Group A.) Clearly though we would not want to have a robot that talked non-stop, we would like to maintain at least *some* interactivity.

Lastly, Figure 4.10 shows the relationship over all interactions in the study of total errors versus user enjoyment. (Again user enjoyment is measured as their agreement level with the sentence ‘I enjoyed talking to the system.’) The graph exhibits a negative correlation, verified by statistical analysis. This confirms the common-sense hypothesis that users enjoy a system better if it makes fewer understanding errors.

---

Figure 4.5: Breakdown of Error Types

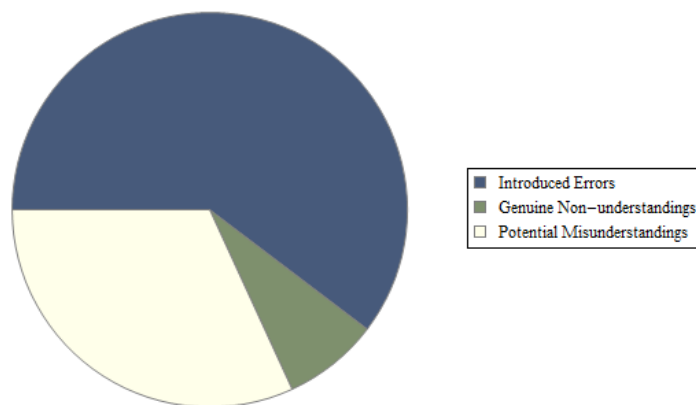
**With Misunderstanding Detection Off**



Total Errors: 112 (per-person average: 8)

---

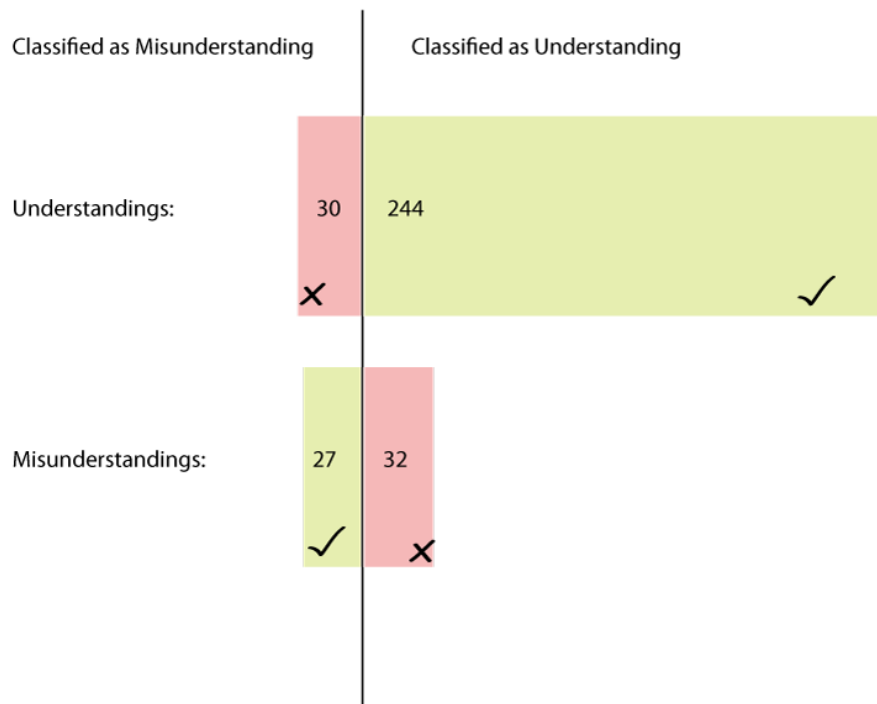
**With Misunderstanding Detection On**



Total Errors: 179 (per-person average: 11.19)

---

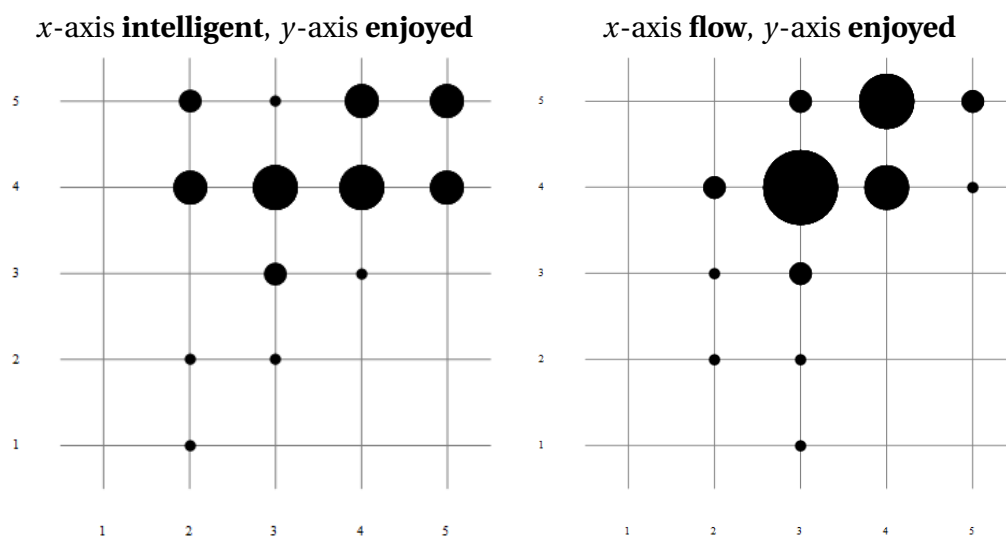
Figure 4.6: Analysis of Misunderstanding Classification



16 dialogues total.

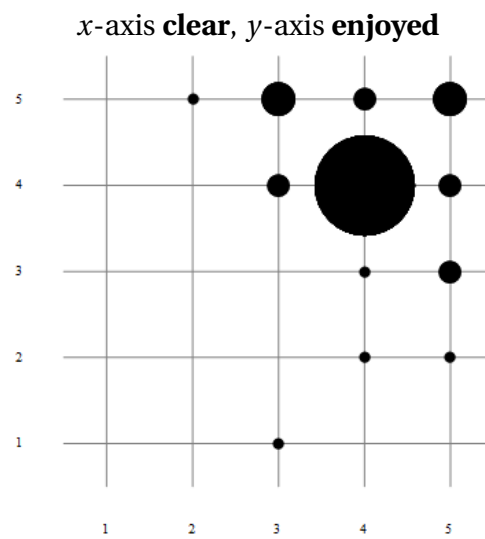
Top left: false positives. Top right: true negatives (estimated from 4 dialogues). Bottom left: true positives. Bottom right: false negatives (estimated from 4 dialogues).

Figure 4.7: Correlation of Questionnaire Questions



---

Figure 4.8: Non-Correlation of Questionnaire Questions

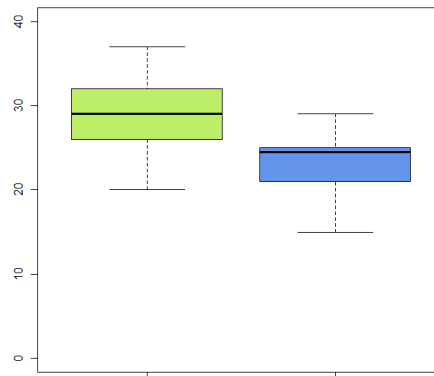




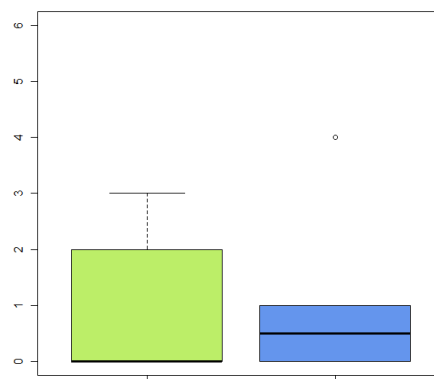
---

Figure 4.9: Total Number of Questions Across Groups

**Number of questions asked of the system**  
(Left: Group A, Right: Group B)



**Number of genuine non-understanding errors**

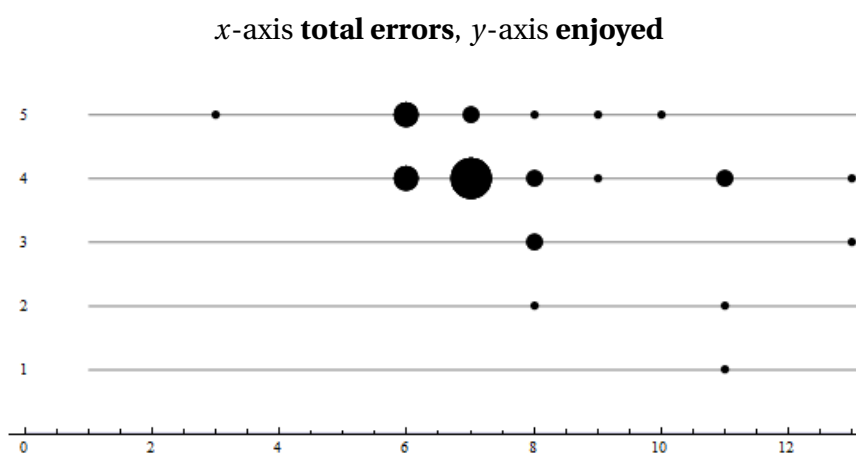


Data from the round where Misunderstanding Detection was switched off.

---

---

Figure 4.10: Negative Correlation of Enjoyment with Error Rate



# Chapter 5

## Conclusions

### 5.1 Conclusions

The Experimental Study provided strong evidence that the new strategies which use the idea of moving on the dialogue can have a positive effect on overall user satisfaction. It was found that in particular the dialogues are made more enjoyable, better flowing, more interesting and less frustrating than the dialogues with a system that only uses the Please Repeat Strategy.

Therefore when designing a non-task-oriented dialogue system, it is certainly worthwhile to put some thought into designing error-recovery strategies which are more complex than asking the user to repeat themselves. This has been confirmed experimentally in task-oriented domains, (see Section 2.4), and this dissertation provides initial supporting evidence in non-task-oriented systems.

The issue of Misunderstanding Detection was also explored. The main finding is that it seems a hard-coded and fixed threshold is not ideal. Misunderstanding Detection had little effect on user opinions, and in fact the classification success was not brilliant. This issue really deserves its own investigation, instead of being an afterthought as it was in this study. More on this in the following Section (5.2).

A rather simple Dialogue System has been presented, just sophisticated enough to test the different error strategies. This has shown that the TrindiKit framework is a very powerful tool for quickly creating dialogue systems. Although the system does not do anything sophisticated in terms of Natural Language Generation and Understanding, the TrindiKit framework would allow for an easy integration of something more complex.

The Tourguide Dialogue System however managed to perform well even with its simple components. Although it isn't possible to directly compare it with any other Dialogue System, it did score towards the top ends of some questions in the questionnaire. For example, it scored

an average of 4.7 out of 5 in level of agreement with 'I enjoyed talking to the system' when the strategies were in use and Misunderstanding Detection (MD) was turned off, and 4.3 with strategies and no MD.

## 5.2 Future Work

Here is presented a list of further related investigations which could be interesting.

- **More strategies.** One idea for another strategy would be using data held by the system on what questions people have asked before, and what questions the current user has asked so far. Perhaps a probabilistic model could be written to try to predict what pieces of information would be interesting to the user using this data, and the results of this could be used to generate subsuming information which is tailored to the user.
- **Strategy comparisons.** The experimental study presented here was only able to compare using the strategies with not using them. It would be interesting to compare within the strategies, to see the strengths and weaknesses of each one. This would necessarily require quite a few more participants in order to achieve statistical significance. It is easier to compare strategies in a slot-filling dialogue, as you can measure how successfully an individual slot was filled when a particular strategy was used. Bohus did something similar to compare a set of strategies in his conference room booking system [1]. However in a non-task oriented domain it is harder to isolate the effect of a single strategy in a dialogue.
- **Strategy policies.** In this study, strategies were chosen at random, using a pseudo random number generator. It may be the case though that it is possible to be more effective if strategies are chosen more carefully. If the effect of individual strategies can be isolated as may be required for the previous point, then it could be possible to learn a model for what error-recovery strategy is most appropriate at every point in the dialogue. Bohus tested the hypothesis that there exists a better-than-random strategy in the conference room booking system by comparing a random choice policy with a policy driven by the choices of a human Wizard [1].
- **Better Misunderstanding Detection.** The MD technique of defining a confidence interval in which to ask for confirmation was shown to only have limited success. A study could be carried out whereby different MD techniques are trialled. The baseline system would be that used in this study. These could look at cues which signal a misunderstanding. Krahmer et al [21] analyse a series of cues which go beyond the classical confidence score techniques, and argue it is possible to use Machine Learning to learn better MD components. Hirasawa et al. also present a set of useful features available to a dialogue

system which could be used to detect misunderstandings [22]. Bohus shows that this is indeed possible, creating a successful model which uses a variety of features including confidence scores, dialogue history, acoustic information and grammatical features [18]. It should be noted though that these ideas for improved Misunderstanding Detection are currently aimed at detecting misunderstandings at the following steps in the dialogue.

The last point is that it would be good to simply gather more data, particularly in an actual human-robot interaction. It is hoped that next year the system will be installed on a moving robot, and will be open to the public to interact with. This will facilitate the collection of natural data, with real errors due to poor acoustics, multiple speakers etc.

# Bibliography

- [1] D. Bohus and A. Rudnicky, “Sorry, I didn’t catch that! An investigation of non-understanding errors and recovery strategies,” in *6th SIGdial Workshop on Discourse and Dialogue*, 2005.
- [2] T. Zollo, “A Study of Human Dialogue Strategies in the Presence of Speech Recognition Errors,” *New York*.
- [3] G. Skantze, “Exploring human error handling strategies: Implications for spoken dialogue systems,” in *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, 2003, pp. 71–76.
- [4] M. O’Donnell, C. Mellish, J. Oberlander, and A. Knott, “ILEX: an architecture for a dynamic hypertext generation system,” *Natural Language Engineering*, vol. 7, no. 03, pp. 225–250, Aug. 2001.
- [5] S. Thrun, M. Bennewitz, W. Burgard, a.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, “MINERVA: a second-generation museum tour-guide robot,” *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 3, pp. 1999–2005. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=770401>
- [6] S. Larsson and D. R. Traum, “Information state and dialogue management in the TRINDI dialogue move engine toolkit,” *Natural Language Engineering*, vol. 6, no. 3&4, pp. 323–340, Sep. 2000.
- [7] S. Young, “Large vocabulary continuous speech recognition: A review,” *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 45–57, 1996.
- [8] J. L. Austin, *How to Do Things With Words*. New York: OUP, 1962.
- [9] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1970.
- [10] Y. W. Wong and R. J. Mooney, “Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus,” *Computational Linguistics*, no. June, pp. 960–967, 2007.

- [11] M. White, "Reining in CCG chart realization," *Natural Language Generation*, pp. 182–191, 2004.
- [12] A. J. Hunt and A. W. Black, "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database," 1996.
- [13] I. Bulyko, K. Kirchhoff, M. Ostendorf, and J. Goldberg, "Error-correction detection and response generation in a spoken dialogue system," *Speech Communication*, vol. 45, no. 3, pp. 271–288, Mar. 2005.
- [14] G. A. Sanders, A. N. Le, and J. S. Garofolo, "EFFECTS OF WORD ERROR RATE IN THE DARPA COMMUNICATOR DATA DURING 2000 AND 2001," *Word Journal Of The International Linguistic Association*, no. 1, 2001.
- [15] T. Koulouri and S. Lauria, "A WOz framework for exploring miscommunication in HRI," in *Procs. of the AISB Symposium on New Frontiers in Human-Robot Interaction*, 2009, pp. 1–8.
- [16] D. Bohus, "Error Awareness and Recovery in Conversational Spoken Language Interfaces," Ph.D. dissertation, 2007.
- [17] S. Konstantopoulos, I. Androutsopoulos, H. Baltzakis, V. Karkaletsis, C. Matheson, A. Tegos, and P. Trahanias, "INDIGO : Interaction with Personality and Dialogue Enabled Robots," *Artificial Intelligence*, pp. 5–6.
- [18] D. Bohus and A. Rudnicky, "Constructing accurate beliefs in spoken dialog systems," *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2005., pp. 272–277, 2005.
- [19] B. Hockey, O. Lemon, E. Campana, L. Hiatt, and Others, "Targeted Help for Spoken Dialogue Systems: intelligent feedback improves naive users' performance," in *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 147–154.
- [20] L. Karsenty and V. Botherel, "Transparency strategies to help users handle system errors," *Speech Communication*, vol. 45, no. 3, pp. 305–324, Mar. 2005.
- [21] E. Krahmer, M. Swerts, M. Theune, and M. Weegels, "Error detection in spoken human-machine interaction," *International journal of speech technology*, vol. 4, no. 1, pp. 19–30, 2001.
- [22] J.-i. Hirasawa, N. Miyazaki, M. Nakano, and K. Aikawa, "New feature parameters for detecting misunderstandings in a spoken dialogue system," 2000.